Amendments to the Specification

Title of the Invention

Please change the title of the invention as follows:

A1 ~~Branch Prediction~~ Method ~~Using~~ of an Address Trace Cache Storing Loop Control Information to Conserve Trace Cache Area

Please amend the paragraph at page 1 lines 7 through 8 as follows:

A2 The present invention relates to a method of storing data in an address trace cache in the environment of branch prediction ~~method and, more particularly, to a branch prediction method using an address trace~~.

Please amend the paragraph at page 3, lines 23 through page 4 line 1, as follows:

A3 A dynamic sequence, which is identical to the discontinuous blocks in the instruction cache 21 shown in Fig. 2A, is continuous in the trace cache 22 shown in Fig. 2B. Therefore, instructions stored in the trace cache 22 can sequentially be executed without repeated branch to an address including an instruction according to a conventionally programmed routine. This makes it possible to prevent a branch penalty which occurs in conventional prediction techniques. ~~And~~ Also, instructions stored in discontinuous positions of the instruction cache 21 are ~~continuously~~ contiguously stored in the trace cache to carry out improved parallel processing.

Please amend the paragraph at page 4, lines 11 through 16 as follows:

A4

According to the aspect of the present invention, a branch prediction method uses a trace cache. If a routine composed of unrepeated instructions is to be executed, an address corresponding to each instruction in the trace cache according to an order of executed instructions is stored. If a routine composed of repeated instructions is carried out, a routine start address, a routine end address, ~~current access times~~ current iteration count of the routine, and ~~total access times~~ total number of iterations of the routine are counted and stored.

Please amend the paragraph at page 4 lines 17 through 21 as follows:

A5

If the routine composed of the repeated instructions is carried out, the trace cache includes loop counters for counting the current ~~access times~~ iteration count and the total ~~current access times~~ number of iterations. If values of the loop counters are identical to each other, a start address that will be subsequent to the routine is addressed. If the branch prediction is missed, the loop counter is recomposed using the latest updated loop count value.

Please amend the paragraph at page 5, lines 1 through 2 as follows:

A6

Fig. 1 contains schematic functional block diagrams which illustrate various microarchitectures including fourth-generation microarchitectures, in accordance with the prior art.

Please amend the paragraph at page 5, lines 3 through 4 as follows:

A7

Fig. 2A is a schematic diagram which shows a dynamic sequence of basic blocks stored in a conventional instruction cache, in accordance with the prior art.

Please amend the paragraph at page 5, line 5 as follows:

A8

Fig. 2B is a schematic diagram which illustrates a conventional trace cache, in accordance with the prior art.

Please amend the paragraph at page 5, lines 6 through 7 as follows:

A9

Fig. 3 is a schematic diagram which illustrates one example of a repetitive instruction pattern, in accordance with the prior art.

At page 5, between lines 13 and 14, please insert the following:

A10

Fig. 7 is a schematic diagram which illustrates one example of a nonrepetitive instruction pattern.

Fig. 8 is a schematic diagram which illustrates a structure of an address trace cache according to the present invention storing the instructions of Fig. 7.

Fig. 9 is a schematic diagram which illustrates a pattern where instructions shown in Fig. 7 are stored in a trace cache according to the invention.

Please amend the paragraph at page 6 lines 8 through 12 as follows:

A11

In Fig. 5, in accordance with the invention, an address trace cache 220 is composed of a start address for storing an address where each routine is started, an end address for an address where each routine is finished, ~~an access current~~ a current iteration loop counter for counting the iterations a ~~access times~~ of a corresponding routine to generate a current iteration count, and ~~an old access~~ a total iteration loop counter for indicating total ~~access times~~ number of iterations of the routine.

Please amend the paragraph at page 6, lines 13 through 20 as follows:

A12

For example, if the information of instructions executed in routine 1 is indicated the address trace cache 220, the start address and the end address of routine 1 are stored in the trace cache 220. Then, current ~~access time~~ iteration count of routine 1 is stored in the current ~~access~~ iteration loop counter while total ~~access times~~ number of iterations (e.g., 30 times) of routine 1 is stored in the ~~old access~~ total iteration loop counter. As ~~access~~ execution of the routine is repeated, a value of the current ~~access~~ iteration loop counter is increased. If the value of the current ~~access~~ iteration loop counter is identical to that of the ~~old access~~ total iteration loop counter, routine 1 is finished and a start address of routine 2 is stored as a next fetch point (NFP).

Please amend the paragraph at page 6 line 27 through page 7 line 4 as follows:

A13

Referring now to Fig. 6, if, for example, a routine 1 shown in Fig. 3 is stored in the address trace cache 220, an address of an initially executed instruction A is stored into a start address of routine 1 while an address of a finally executed instruction B is stored into an end address thereof. Since the total number of iterations or repetitions ~~times~~ of the routine 1 is 30, ~~an old access~~ the value of the total iteration loop counter is stored as 30. Whenever routine 1 is repeatedly carried out, a value of [[a]] the current ~~access~~ iteration loop counter increases by 1. In the same manner, information of routines 2 and 3 is stored in the address cache 220.

Please amend the paragraph at page 7 lines 5 through 12 as follows:

A14

As shown in Fig. 6, since the address cache 220 is composed of an address where each routine is started, an address where each routine is finished, a current ~~access~~ iteration loop counter, and ~~an old access~~ a total loop counter, only four data storing areas are required to store information of a repeated routine. Therefore, total 12 data storing areas are required to store routines 1 through 3 in the address trace cache 220. In this case, if 32 bits are utilized to store each piece of information, a total of 384 bits (i.e., 48 bytes) are required to store routines 1 through 3. This is smaller by about 16.7 times than a data storing area utilized in a conventional trace cache.

Please insert at page 7, after line 12, the following new paragraph:

A15

Figs. 7 through 9 illustrate handling instructions in a routine that is not to be repeated in accordance with the invention. Figs. 7 through 9 are analogous to Figs. 3, 5 and 6, respectively, which illustrate handling of repeated instructions. Fig. 7 is a schematic diagram which illustrates one example of a nonrepetitive instruction pattern. Fig. 8 is a schematic diagram which illustrates a structure of an address trace cache according to the present invention storing the instructions of Fig. 7. Fig. 9 is a schematic diagram which illustrates a pattern where instructions shown in Fig. 7 are stored in a trace cache according to the invention.

Please amend the Abstract of the Disclosure at page 9 lines 1 through 9 as follows:

A16

A branch prediction method using an address trace is described. An address trace corresponding to an executed instruction is stored itself with a decoded form. After appointing a start address and an end address of a repeated routine, current routine accessing times and total accessing times are compared with each other, confirming the end of the routine and storing address information of the next routine. Therefore, access information of the repeated routine can be stored using a small amount of a trace cache.

~~J:\SAM\0169\169patapp2.wpd~~